第 37 卷第 4 期
录用定稿
计算机应用研究
Application Research of Computers
Vol. 37 No. 4
Accepted Paper

异构容器云间应用迁移模型研究 *

杨凯琪，赵玉龙，陈　林
(江南计算技术研究所，江苏 无锡 214000)

摘　要：为解决基于 Docker 容器的应用跨异构容器云迁移的问题，针对主流的异构容器编排引擎的编排原理进行了研究，同时也研究了主流容器服务提供商之间的异构性，在此基础上提出了基于 Docker 容器的应用跨异构容器云迁移的三层模型。为提高基于 Docker 容器的应用跨异构容器云迁移的效率，提出基于镜像预同步的应用迁移技术。实验结果表明，利用三层模型能成功实现应用跨异构容器云的迁移，并且在引入镜像预同步技术前后，同构云异构容器编排引擎之间的应用迁移时间平均减少 60.33%，异构云 Kubernetes 集群之间的应用迁移时间平均减少 43.67%。
关键词：Docker 容器；应用迁移；三层模型；异构容器云
中图分类号：TP302　　doi: 10.19734/j.issn.1001-3695.2018.09.0762

Study on migration of applications between Heterogeneous container clouds

Yang Kaiqi, Zhao Yulong, Chen Lin
(*Institute of JiangNan Computing Technology*, *Wuxi* Jiangsu 214000, *China*)

Abstract: In order to migrate Docker container based applications across heterogeneous container clouds, the author studied the principles of mainstream heterogeneous container orchestrating engines. Meanwhile, this paper conducted a survey on the heterogeneity of container service of different providers. The author proposed the Three-layered Model of Migrating Docker Container based Applications across Heterogeneous Container Clouds. Four the purpose of improving migration efficiency, this paper developed the Image Pre-Synchronization technology. Experimental results show that using the three-layered model can migrate Docker container based applications across container clouds successfully, and the technology of image pre-synchronization can help the time of migrating. For the situation of migrating Docker container based applications between heterogeneous container orchestrating engines in homogeneous clouds, the improvement is 60.33% on average. And for the situation of migrating Docker container based applications between Kubernetes clusters in heterogeneous clouds, the improvement is 43.67% on average.
Key words: docker containers; migration of applications; three-layered model; Heterogeneous container clouds

0　引言

企业总是运行多种异构技术。目前云计算服务市场处于百花齐放、百家争鸣的状态，国外的云服务提供商包括亚马逊、微软、谷歌等，国内的云服务提供商包括阿里、华为、美团等。为了充分利用就近计算优势，提高业务应用的全域服务能力以及避免被特定的云计算服务提供商锁定，企业需要将自己的业务应用部署到多个地域、多个云计算服务提供商的数据中心内[1]。但是在跨域传输数据时需要付出很大的代价：一方面，跨域数据传输不仅消耗时间长，而且消耗宝贵的网络带宽资源，性价比低；另一方面，安全规则限制下某些数据不允许被跨域传输。因此现在学术界和工业界都提倡在异构容器云之间进行应用的迁移。

然而现阶段异构容器云间的应用迁移仍然存在着许多挑战：a)对于复杂应用来说，如何在迁移过程中保持与编排引擎的交互，从而维持应用在迁移前后的健康检查机制、节点亲和性等编排规则的一致性；b)基于 Docker 容器的应用如何抛开迁移前后的存储和网络环境能满足容器应用运行所需的条件；c)如何让基于 Docker 容器的应用总迁移时间更短；d)频繁的迁移操作会导致数据中心的能耗快速增加，以及占用网络带宽，对数据中心的日常使用带来影响。

目前对多数据中心的容器管理主要是以云联邦（cloud federation）的形式进行[1~3]，如 Kubernetes 集群联邦、Hashicorp Nomad 和 Mesos 集群联邦等。虽然它们都能对多个数据中心的容器进行管理，但都局限于同构容器编排引擎，无法调度异构容器编排引擎之间的资源，不能将基于 Docker 容器的应用在 Kubernetes 集群[10]与 Docker Swarm 集群[11~13]之间调度。文献[4]中提出了基于日志重放的容器迁移方法，该方法的迁移时间随着容器运行时间的增加而增加，然而企业级应用往往需要长时间运行，因此该方法存在耗时长的问题，而且该方法仅适用于同构的容器云环境；文献[5,6]中利用 Checkpoint-Restart 技术来进行容器应用迁移，但是该方法需要迁移前后的环境中文件系统一致，并且要求网络一致，无法适用于不同云提供商、跨地域的多数据中心的应用迁移。国外主流的研究 Docker 容器迁移技术团队 CRIU[8]也是利用了与 CR 技术类似的思想来进行容器迁移；文献[7]中提出的应用迁移三层模型将应用迁移过程分为 Base Layer、Application Layer 和 Instance Layer 这三层，该方法虽然能在 Base Layer 层保证迁移前后文件系统一致性，但是由于未考虑到异构容器云之间的异构性，无法实现迁移前后对应用的接管。因此文献[4~7]中提出的方法同样都无法解决基于 Docker 容器的应用在异构容器云之间的迁移。

收稿日期：2018-09-24；**修回日期：**2018-11-08　**基金项目：**国家重点研发计划资助项目（2016YFB1000505）
作者简介：杨凯琪（1993-），男，湖南衡山人，硕士研究生，主要研究方向为云计算、容器技术（1851206559@qq.com）；赵玉龙（1984-），男，安徽临泉人，工程师，硕士，主要研究方向为云计算、容器技术等；陈林（1977-），女，湖北武人，高级工程师，硕士，主要研究方向为云操作系统.

本文提出基于 Docker 容器的应用跨异构容器云迁移三层模型 TMHCC，该模型能同时适用于同构云异构容器编排引擎和异构云中 Kubernetes 集群的情况。通过镜像预同步技术的引入，能有效提高应用迁移的效率。

1　基于 Docker 容器的应用跨异构容器云迁移三层模型 TMHCC

1.1　异构容器云定义

异构容器云分为同构云异构容器编排引擎和异构云同构容器编排引擎两种情形。前者是指云平台本身的是同构的，但是容器服务使用了不同的容器编排引擎。而后者则是指虽然云平台本身是异构的，但是使用的都是同一种容器编排引擎。若两个数据中心都使用的是阿里云的容器服务，两者都使用相同类型的存储和网络，但是一个数据中心使用的是 Kubernetes 容器编排引擎，而另一个数据中心使用的是 Docker Swarm 容器编排引擎，这种情形就是同构云异构容器编排引擎。若两个数据中心一个使用的是阿里云的容器服务，而另一个数据中心使用的是华为云的容器服务，两家容器服务在网络和存储方面存在着异构性，但是两个数据中心用的都是 Kubernetes 容器编排引擎，这种情形就是异构云同构容器编排引擎。

1.2　三层模型

基于 Docker 容器的应用跨异构容器云迁移三层模型如图 1 所示。该模型将应用在进行跨容器云迁移时分成镜像层、编排层和应用层这三个层次。

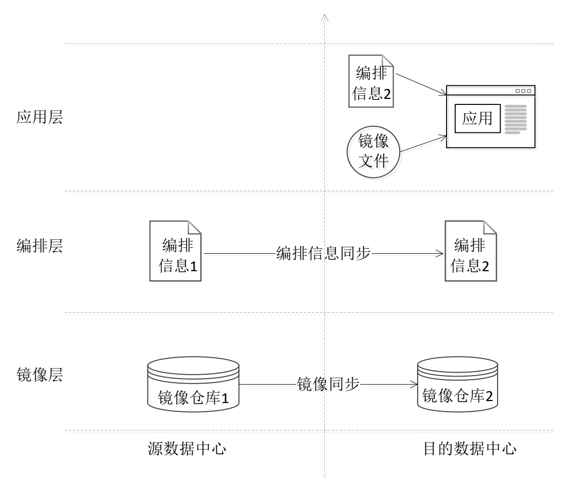


图 1　基于 Docker 容器的应用跨异构容器云迁移三层

Fig. 1　Three-layered model of migrating Docker container based applications across heterogeneous container clouds

1.2.1 TMHCC 镜像层

在 Docker 中，容器镜像是容器的基础[9]。因此在将基于 Docker 容器的应用进行迁移时，确保迁移前后的源数据中心和目的数据中心之中含有该应用所依赖的镜像文件至关重要。基于 Docker 容器的应用跨云迁移三层模型在镜像层同步源数据中心和目的数据中心的镜像仓库。Docker、Kubernetes 以及各云提供商（如华为和阿里）等都遵循 OCI 规范，因此源数据中心和目的数据中心的容器镜像格式都相同。在进行镜像同步时无须对镜像格式进行转换。

1.2.2 TMHCC 编排转换技术

基于 Docker 容器的应用跨云迁移三层模型在编排层根据源数据中心和目的数据中心的特性进行编排模板的同步。应用的编排模板包含一组容器服务的定义及其相互关联，可用于多容器应用的部署及管理。Docker Swarm 支持 Docker Compose 模板规范，可以用 Compose 文件来编排和部署应用。Kubernetes 集群支持以 YAML 文件或者 JSON 文件的格式来编排和部署应用的各种资源对象，也支持通过 Helm 来简化应用的部署。对于应用在同构云异构容器编排引擎之间迁移的情形 ，编排层通过 SwarmNetes 来同步编排模板；对于应用在异构云 Kubernetes 集群之间迁移的情形，编排层通过 HelmConvert 来同步编排信息 Chart 模板。

对于同构容器云中的 Docker Swarm 集群和 Kubernetes 集群之间的应用迁移，通过设计和研发 SwarmNetes 来实现编排信息同步。SwarmNetes 架构如图 2 所示。SwarmNetes 能将 v1、v2 和 v3 版本的 Docker Compose 编排文件与 Kubernetes 的 ConfigMap、Service、ReplicationController、DaemonSet、Deployment、Pod 和 PersistentVolumeClaim 这 6 中资源对象进行相互转换。

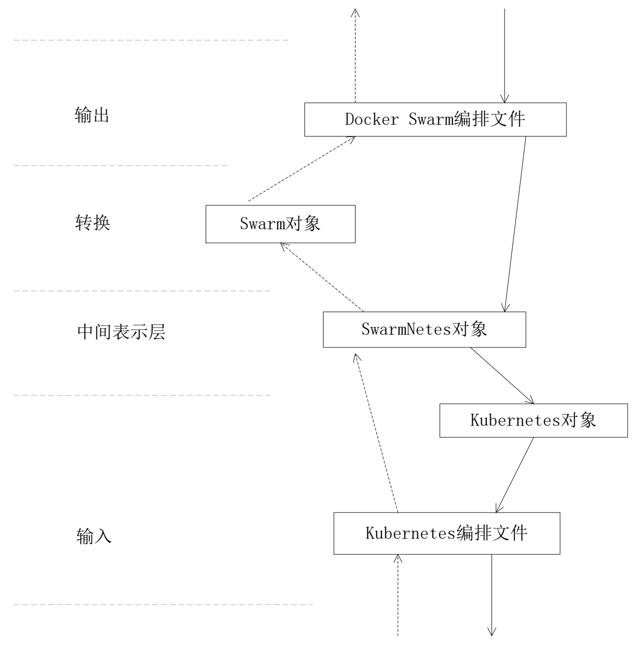


图 2　SwarmNetes 架构

Fig. 2　Architecture of swarmnetes

虚线所示方向为应用从 Kubernetes 集群向 Docker Swarm 集群迁移时编排文件转换过程，SwarmNetes 从输入的 Kubernetes 编排信息文件中提取应用信息，并保存在中间表示层 SwarmNetes 对象中，通过转换将 SwarmNetes 对象转换为 Docker Swarm 集群可以识别的 Swarm 对象，最后根据 Swarm 对象输出 Docker Swarm 集群的编排文件；实线所示方向为应用从 Docker Swarm 集群向 Kubernetes 集群迁移时编排文件转换过程，SwarmNetes 从输入的 Docker Swarm 编排信息文件中提取应用信息，并保存在中间表示层 SwarmNetes 对象中，通过转换将 SwarmNetes 对象转换成 Kubernetes 可以识别的 Kubernetes 对象，最后根据 Kubernetes 对象输出 Kubernetes 集群的编排文件。SwarmNetes 实现编排信息转换算法如算法 1 所示。

算法 1 SwarmNetes 编排信息转换算法

```
1. SwarmNetesObject=LoadFile(InputFiles);
2. if Provider == Kubernetes then
3. SwarmObject=kubernetes.Transform(SwarmNetesObject)
4. SwarmFile=Kubernetes.Output(SwarmObject)
5. else
6. KubernetesObject=swarm.Transform(SarmNetesObject)
7. KubernetesFile=Swarm.Output(KubernetesObject)
```

```
8. end if
```

由于异构云存储和网络方面存在的异构性，使得基于 Docker 容器的应用在异构云的 Kubernetes 集群之间迁移时存在无形的障碍，即若一个基于 Dokcer 容器的应用使用了某个云服务提供商特定的存储或者网络，迁移到基于另一个云服务提供商的数据中心时，该应用无法成功启动，目的数据中心无法识别应用中存储和网络的类型。对于异构容器云 Kubernetes 集群之间的应用迁移，本文通过设计和研发 HelmConvert 将源数据中心应用的 Chart 模板转换为适用于目的数据中心的 Char 模板。

HelmConvert 的架构如图 3 所示。输入层读取一个应用的 Chart 模板，并对读取后的内容进行相应的处理，如将 Chart 模板中 values.yaml 文件中定义的各项配置渲染到 templates 文件夹中定义的各种资源对象等。识别层对输入层的结果进行识别，该层主要根据用户提供的云服务提供商类别，如“Huawei”或者“Ali”，分别表示华为容器服务提供商以及阿里容器服务提供商，以及各云服务提供商容器服务的特性来识别输入的 Chart 模板使用了何种类型的存储和网络。在转换层，主要根据本文给出的异构存储和网络转换方法进行相应的转换，使得转换后的 Chart 模板能适用于目的数据中心的容器服务提供商。最后在输出层对转换后的 Chart 模板进行输出。

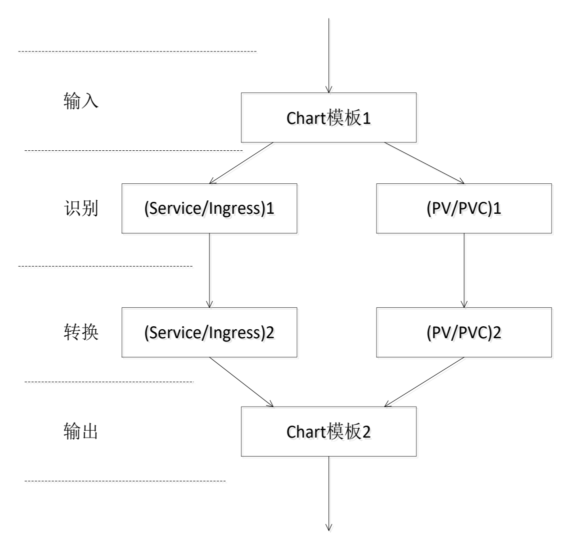


图 3　HelmConvert 架构

Fig. 3　Architecture of helmconvert

华为云容器服务支持 EVS（Elastic Volume Service，云硬盘）、文件存储和 OBS（Object Storage Service，对象存储服务等类型的存储，阿里云容器服务支持阿里云云盘、NAS（Network Attached Storage，阿里云文件存储）、OSS（Object Storage Service，阿里云对象存储）这三种类型的存储。虽然各家在各种存储实现方式上存在着区别，但是华为云容器服务和阿里云容器服务都支持以 PV（持久化存储卷 PersistentVolume）和 PVC（持久化存储卷声明 PersistentVolumeClaim）的形式来提供存储资源。Kubernetes 提供 PV 和 PVC 这两个 API 资源来抽象存储的细节。集群管理员只需关注如何通过 PV 提供存储功能，而无须关注用户如何使用它。类似的，用户仅需将 PVC 挂载到容器中，而不需关注存储卷的具体技术实现。存储的异构性对用户是透明的。PVC 和 PV 的关系与 Pod 和 Node 的关系类似，前者消耗后者的资源。正是由于 PV 和 PVC 对存储细节的抽象，使得异构存储之间相互转换有了可行的解决方法。HelmConvert 将华为云 EVS 和阿里云云盘互相转换、华为云文件存储和阿里云 NAS 互相转换以及华为云 OBS 和阿里云对象存储互相转换。

容器服务为满足多种复杂场景下基于 Docker 容器的应用间的互相访问，提供了不同的访问方式，从而满足不同场景提供不同的访问通道。这些网络访问可以分为集群内访问、VPC 内访问、四层负载均衡和七层负载均衡这四种访问方式。集群内房网和 VPC 内访问属于私网访问，后两者为公网访问。在研究中发现，对于集群内访问方式，华为云和阿里云都遵循原生 Kubernetes 标准，因此当基于 Docker 容器的应用使用集群内访问方式时，可以在异构云平台之间进行无缝迁移。对于其他三种访问方式 HelmConvert 通过特征属性识别相应网络类型并进行转换。

1.2.3 TMHCC 应用层

基于 Dokcer 容器的应用依赖于镜像文件以及对这个应用的编排信息文件。源数据中心的应用经过在镜像层对镜像文件的同步和在编排层对编排信息的同步，目的数据中心已经具备该应用的基本要素。最后在应用层根据编排信息文件和镜像文件启动该应用即可。对于同构云异构容器编排引擎的情况，当目的数据中心为 Kubernetes 集群时，通过 kubectl create －f [file]可以创建相应的应用；当目的数据中心为 Docker Swarm 集群是，通过 docker-compose up 或者 docker stack deploy －compose-file=filename 可以创建相应的应用。对于异构云 Kubernetes 集群的情况，通过 Helm 来简化了应用的管理，利用 helm install 指令即可创建应用。

1.3　基于镜像预同步的应用迁移技术

对于未采用镜像预同步技术进行应用迁移的方法本文称之为传统方法。传统方法如图 4 所示，需要进行以下五个步骤：

a)将源数据中心应用的编排信息文件 1 转换为目的数据中心能识别的编排信息文件 2，并将编排信息文件 2 同步到目的数据中心。在该步中，根据不同情形用 SwarmNetes 或者 HelmConvert 进行相应的转换；

b)在目的数据中心根据编排信息文件 2 创建应用时会从镜像仓库 2 拉取应用所依赖的镜像文件，若镜像仓库 2 中没有该应用所依赖的镜像文件，则进行 step3；若镜像仓库 2 中有该应用所依赖的镜像文件，则进行 step4；

c)从源数据中心的镜像仓库 1 同步应用所需的镜像文件到镜像仓库 2 中；

d)从镜像仓库 2 中拉取应用所需的镜像文件；

e)根据编排信息 2 和镜像文件创建运行并运行。

传统方法实现应用迁移所需的总时间 T_1 如式(1)所示。

$$T_1 = T_{OrcSync} + T_{ImageSync} + T_{APPRun} \tag{1}$$

$$T_{OrcSync} = T_{OrcConvert} + T_{OrcRep} \tag{2}$$

其中：$T_{OrcSync}$ 为编排信息同步的时间，是 $T_{OrcConvert}$ 和 T_{OrcRep} 之和。$T_{OrcConvert}$ 为 SwarmNetes 在源数据中心将编排信息 1 转换为编排信息 2 的时间。对于同构云异构容器编排引擎的情形，$T_{OrcConvert}$ 即为通过 SwarmNetes 进行编排信息转换的时间；对于异构云 Kubernetes 集群的情形，$T_{OrcConvert}$ 即为 HelmConvert 进行编排信息转换的时间。T_{OrcRep} 为将编排信息 2 从源数据中心同步到目的数据中心所用的时间。$T_{ImageSync}$ 为镜像仓库 1 和镜像仓库 2 进行镜像同步的时间，T_{AppRun} 在目的数据中心通过编排信息 2 和镜像文件创建应用所用的时间。

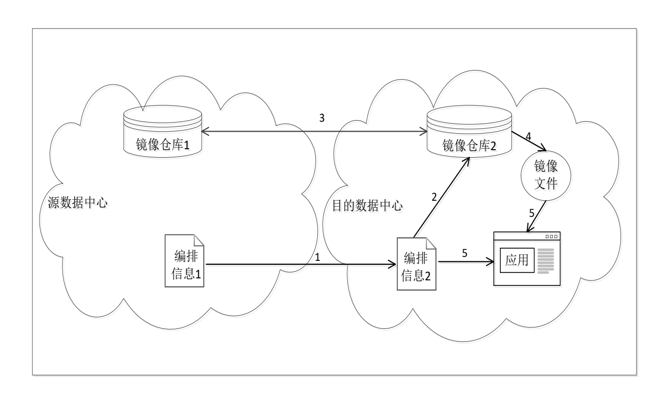


图 4　应用迁移传统方法

Fig. 4　Traditional method of migrating applications

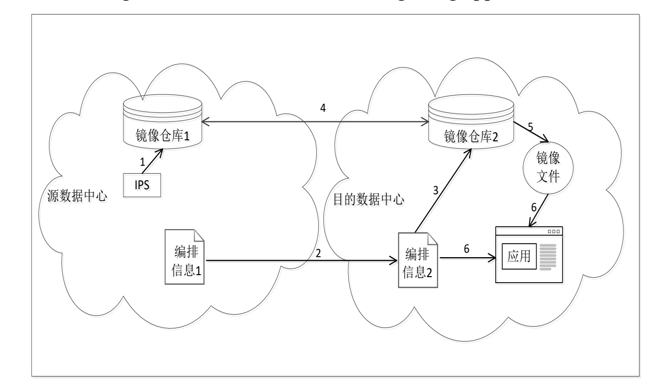


图 5　基于镜像预同步的应用迁移技术

Fig. 5　Technology of migrating applications based on Image Pre-Synchronization

基于镜像预同步的应用迁移技术在源数据中心引入镜像预同步（Image Pre-Synchronization, IPS）模块，IPS 实时监测源数据中心新建镜像的操作，若监测到有新建的镜像文件，立即监测目的数据中心的镜像仓库 2 中是否有该镜像文件，若没有则将该镜像文件同步到镜像仓库 2 中，如图 5 所示。

基于镜像预同步的应用迁移技术需要进行以下步骤：

a)在源数据中新实时检测新镜像文件的生成，若发现新建立的镜像文件，立即同步到目的数据中心，主要由镜像预同步 IPS 模块负责；

b）将源数据中心应用的编排信息文件 1 转换为目的数据中心能识别的编排信息文件 2，并将编排信息文件 2 同步到目的数据中心。在该步中，根据不同情形用 SwarmNetes 或者 HelmConvert 进行相应的转换；

c)在目的数据中心根据编排信息文件 2 创建应用时会从镜像仓库 2 拉取应用所依赖的镜像文件，若镜像仓库 2 中没有该应用所依赖的镜像文件，则进行 step4；若镜像仓库 2 中有该应用所依赖的镜像文件，则进行 step5；

d)从源数据中心的镜像仓库 1 同步应用所需的镜像文件到镜像仓库 2 中；

e)从镜像仓库 2 中拉取应用所需的镜像文件；

f)根据编排信息 2 和镜像文件创建并运行应用。

基于镜像预同步的应用迁移技术实现应用迁移需要的总时间 T_2 如式（3）所示。

$$T_2 = T_{OrcSync} + T_{APPRun} \tag{3}$$

应用迁移时镜像同步阶段占用了主要之间，从 T_1、T_2 的表达式可以看出基于镜像预同步的应用迁移技术大大缩短了基于 Docker 容器的应用进行跨容器云迁移时的总时间。

2　实验与分析

基于镜像预同步的应用迁移三层模型是在基于 Docker 容器的应用跨异构容器云迁移三层模型的基础之上引入了基于镜像预同步的应用迁移技术。本文选取不同的基于 Docker 容器的应用来进行实验，并对引入镜像预同步技术前后的实验结果进行了对比分析。

2.1　同构云异构容器编排引擎

为验证基于镜像预同步的应用迁移三层模型在 Docker Swarm 集群和 Kubernetes 集群之间（即同构云异构容器编排引擎之间）的可行性及有效性。本文选取 Docker 容器中的四个典型应用来验证基于镜像预同步的应用迁移三层模型在同构云异构容器编排引擎情形下的可行性和有效性。这个四个应用分别为：

a)Counter。Counter 应用统计访问数据库的次数，并在 Web 页面上显示。该应用依赖 redis:3.0 和 tuna/docker-counter23:latest 这两个镜像，镜像文件大小分别为 91.5MB 和 696.0MB。

b)Wordpress。Wordpress 是一款个人博客系统，使用 PHP 和 MySQL 数据库开发。该应用依赖 wordpress:4.8-apache 和 mysql:5.6 这两个镜像，镜像文件大小分别为 408.0MB 和 256.0MB。

c)Gitlab。Gitlab 应用用于仓库管理，使用 Git 作为代码管理工具，并在此基础上搭建 web 服务。该应用依赖 redis:latest、postgresql:9.5 和 gitlab:8.13 这三个镜像，镜像文件大小分别为 133.0MB、232.0MB 和 770MB。

d)Vote。Vote 负责统计对 Dog 和 Cat 这两个对象的投票结果，并在 web 页面上显示。该应用依赖 redis:alpha、postgres:9.4、docker/example-voting-app-worker、docker/example-voting-app-vote 和 tmadams333/example-voting-app-result 这 5 个镜像，镜像文件大小分别为 27.8 MB、263.0 MB、574.0 MB、83.5 MB 和 226.0 MB。

2.1.1 实验环境

在该实验中，实验平台选取开源的 Openstack，其版本为 pike。并在 Openstack 中分别建立基于 Kubernetes 集群的 Docker 容器环境和基于 Docker Swarm 的 Docker 容器环境。如图 6 所示，Kubernetes 集群包括 Master 节点（IP 地址为 10.33.122.77）、节点 1（IP 地址为 10.33.122.86）和节点 2（IP 地址为 10.33.122.81）这三个节点，各节点配置参数如表 1 所示；同样 Docker Swarm 集群也包括管理节点（IP 地址为 10.33.35.35），节点 1（IP 地址为 10.33.35.34）和节点 2（IP 地址为 10.33.35.19）这三个节点，各节点参数配置如表 2 所示。各个节点之间的网络带宽均为 1000M。

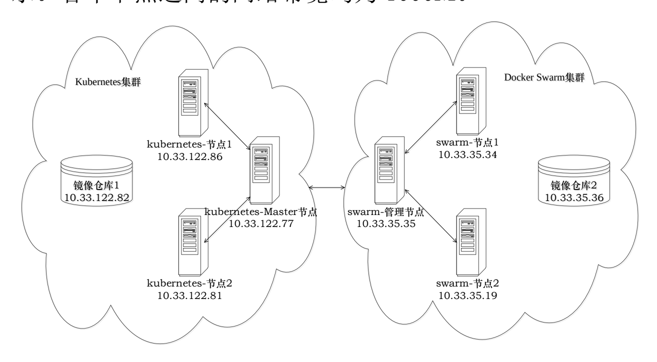


图 6　Kubernetes 集群和 Docker Swarm 集群

Fig. 6　Kubernetes clusters and Docker Swarm clusters

表 1　Kubernetes 集群节点配置

Table 1　Configuration of nodes in Kubernetes clusters

项目	规格说明
操作系统	CentOS7.4 内核版本 3.10
虚拟内核数	2
内存	4GB
跟磁盘	100GB
Docker 版本	17.03.2-ce
Kubernetes 版本	V1.9.0

表 2　Docker Swarm 集群节点配置

Table 2　Configuration of nodes in Docker Swarm clusters

项目	规格说明
操作系统	CentOS7.3　内核版本 3.10
虚拟内核数	4
内存	4GB
跟磁盘	100GB
Docker 版本	17.03.2-ce

2.1.2 实验结果

实验中分别用传统方法和基于镜像预同步的三层模型将 Counter、Wordpress、Gitlab 和 Vote 这四个应用从 Kubernetes 集群迁移到 Docker Swarm 集群，迁移所用时间如图 7 所示。传统方法迁移所用时间分别为 121.71 s、117.46 s、193.25 s 和 185.10s，而利用基于镜像预同步的三层模型进行迁移分别只需要 46.97 s、54.28 s、62.06 s 和 82.08 s。

随后，本文再分别用传统方法和基于镜像预同步的三层模型将 Counter、Wordpress、Gitlab 和 Vote 这四个应用从 Docker Swarm 进群迁移到 Kubernetes 集群之中，迁移所用时间如图 8 所示。传统方法所用时间分别为 128.34 s、120.25 s、184.02 s 和 191.22 s，而利用基于镜像预同步的三层模型进行迁移分别只需要 45.72 s、53.32 s、62.73 s 和 79.52 s。

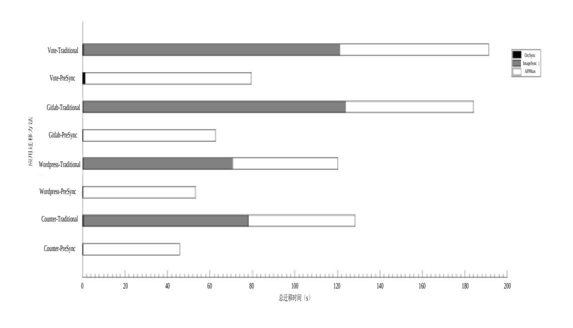


图 7　应用从 Kubernetes 集群迁移到 Docker Swarm 集群所用时间

Fig. 7　Time of migrating applications from Kubernetes clusters to Docker Swarm clusters

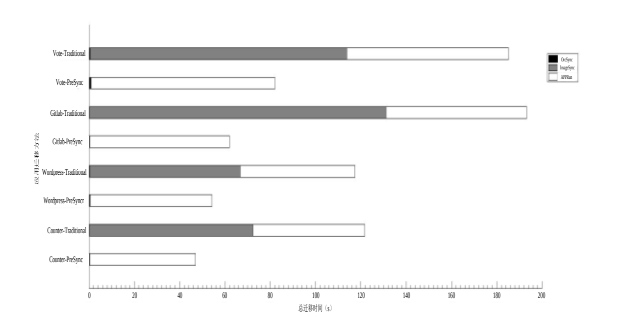


图 8　应用从 Docker Swarm 集群迁移到 Kubernetes 集群所用时间

Fig. 8　Time of migrating applications from Docker Swarm clusters to Kubernetes clusters

从图 7 和 8 的实验结果之中可以看出基于 Docker 容器的应用在在 Docker Swarm 集群和 Kubernetes 集群之间迁移时镜像同步占用了大部分时间，基于镜像预同步的三层模型通过镜像预同步技术在很大程度上减少了应用在 Docker Swarm 集群和 Kubernetes 集群之间迁移的时间，使得应用迁移时间性能平均提升了 60.33%。

2.2　异构云 Kubernetes 集群

为验证基于镜像预同步的应用迁移三层模型在华为云容器服务的 Kubernetes 集群和阿里云容器服务的 Kubernetes 集群之间（即异构云 Kubernetes 集群之间）的可行性和有效性。本文选取 Nginx 这个应用来进行验证。Nginx 由俄罗斯的程序设计师 Igor Sysoev 开发，是一个高性能的 HTTP 服务器、反向代理服务器以及电子邮件代理服务器（IMAP/POP3）。为了尽可能涵盖华为云容器服务和阿里云容器服务之间异构存储和异构网络的所有情况，本文将 Nginx 这个应用分别设计为使用不同类型的存储和网络。应用 Nginx1 在华为云容器服务中使用的存储类型为华为云 EVS，网络类型为 VPC 内访问，在阿里云容器服务中使用的存储类型为阿里云云盘，网络类型为 VPC 内访问；应用 Nginx2 在华为容器服务中使用的存储类型为华为云文件存储，网络类型为四层负载均衡，在阿里云容器服务中使用的存储类型为阿里云 NAS，网络类型为四层负载均衡；应用 Nginx3 在华为云容器服务中使用的存储类型为华为云 OBS，网络类型为七层负载均衡，在阿里云容器服务中使用的存储类型为阿里云 OSS，网络类型为七层负载均衡。Nginx1、Nginx2 和 Nginx3 这三个应用都依赖 nginx:latest 这个镜像，该镜像大小为 109MB。

2.2.1 实验环境

本文在华为云容器服务和阿里云容器服务中分别搭建了 Kubernetes 集群，如图 9 所示。

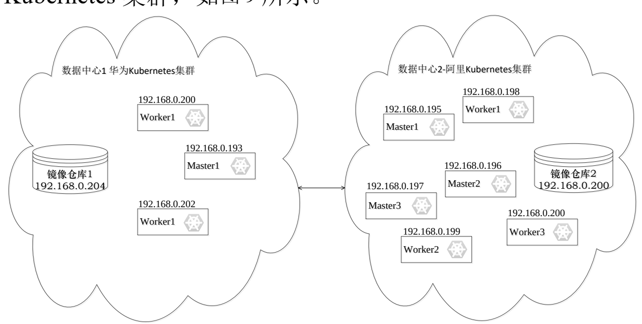


图 9　华为容器服务 Kubernetes 集群和阿里容器服务 Kubernetes 集群

Fig. 9　Kubernetes clusters based on Huawei container service and Kubernetes clusters based on Ali container service

数据中心 1 位于北京，是在华为容器服务中搭建的 Kubernetes 集群，其包括一个 Master 节点（IP 地址为 192.168.0.193）和两个 Worker 节点（IP 地址分别为 192.168.0.200 和 192.168.0.202），并为数据中心 1 搭建了私有镜像仓库 1，其 IP 地址为 192.168.0.204。数据中心 2 位于深圳，是在阿里容器服务中搭建的 Kubernetes 集群，该 Kubernetes 集群拥有过 3 个 Master 节点（IP 地址分别为 192.168.0.195、192.168.0.196 和 192.168.0.197）和 3 个 Worker 节点（IP 地址分别为 192.168.0.198、192.168.0.199 和 192.168.0.200），并为数据中心 2 搭建了私有镜像仓库 2，其 IP 地址为 192.168.0.200。数据中心 1 和数据中心 2 中各节点配置参数如表 3 和 4 所示。

表 3　数据中心 1 各节点参数配置

Table 3　Configuration of nodes in data center 1

项目	规格说明
操作系统	EulerOS 2.0 (SP2)　内核版本 3.10
虚拟内核数	4
内存	8GB
磁盘	40GB（系统盘）+100GB（数据盘）
Docker 版本	17.06.0.10
Kubernetes 版本	v1.9.7
所在区域	华北-北京

表 4　数据中心 2 各节点参数配置

Table 4　Configuration of nodes in data center 2

项目	规格说明
操作系统	CentOS7.4　内核版本 3.10
虚拟内核数	4
内存	8GB
磁盘	40GB（系统盘）
Docker 版本	17.06.2-ce-3
Kubernetes 版本	v1.10.4
所在区域	华南-深圳

2.2.2 实验结果

在验证基于镜像预同步的三层模型在异构云 Kubernetes 集群之间的适用性时，本文分别用传统方法和基于镜像预同步的三层模型将 Nginx1、Nginx2、Nginx3 这三个应用从华为容器服务的 Kubernetes 集群中迁移到阿里容器服务的 Kubernetes 集群之中，迁移所用时间如图 10 所示。在用传统方法进行迁移时，Nginx1、Nginx2 和 Nginx3 这三个应用迁移所用时间分别为 28.16s、24.13s 和 25.96s。而当用基于镜像预同步的应用三层模型进行迁移时，Nginx1、Nginx2 和 Nginx3 这三个应用迁移所用的时间分别只需要 13.72s、12.76s 和 12.95s。

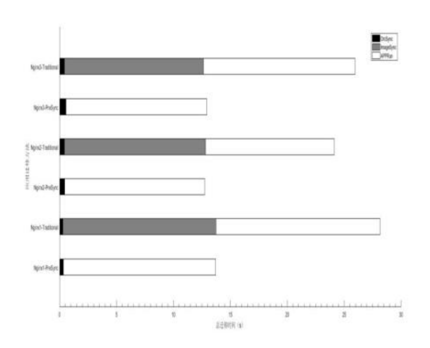


图 10　应用从华为 Kubernetes 集群迁移到阿里 Kubernetes 集群所用时间

Fig. 10　Time of migrating applications from Kubernetes clusters based on Huawei to Kubernetes clusters based on Ali

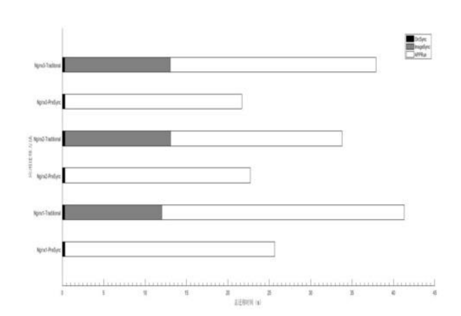


图 11　应用从阿里 Kubernetes 集群迁移到华为 Kubernetes 集群所用时间

Fig. 11　Time of migrating applications from Kubernetes clusters based on Ali to Kubernetes clusters based on Huawei

随后，本文再分别用传统方法和基于镜像预同步的三层模型将 Nginx1、Nginx2 和 Nginx3 这三个应用从阿里容器服务的 Kubernetes 集群中迁移到华为容器服务的 Kubernetes 集群之中，迁移所用时间如图 11 所示。在用传统方法进行迁移时，Nginx1、Nginx2 和 Nginx3 这三个应用迁移所用时间分别为 41.33 s、33.81 s 和 37.91 s。而当用基于镜像预同步的应用迁移三层模型进行迁移时，Nginx1、Nginx2 和 Nginx3 这三个应用迁移所用的时间分别只需要 25.68 s、22.71 s 和 21.68 s。

从图 10 和 11 中的实验结果之中可以看出，基于 Docker 容器的应用在华为容器服务的 Kubernetes 集群和阿里容器服务的 Kubernetes 集群之间迁移时，与基于 Docker 容器的应用在 Docker Swarm 集群和 Kubernetes 集群之间迁移时一样，镜像同步阶段占用了应用迁移的大部分时间。基于镜像预同步的三层模型通过镜像预同步技术在很大程度上减少了应用在华为容器服务的 Kubernetes 集群和阿里容器服务的 Kubernetes 集群之间迁移的时间，使得应用迁移时间性能平均提升了 43.67%。

3　结束语

为满足将基于 Docker 容器的应用在异构容器云之间迁移的需求，本文提出了基于 Docker 容器的应用跨异构容器云迁移三层模型。并在编排层设计和研发了 SwarmNetes 来屏蔽异构容器编排引擎之间的异构性，使得基于 Docker 容器的应用在同构云异构容器编排引擎之间的迁移成为可能；同时也在编排层设计和研发了 HelmConvert 来屏蔽不同提供商的容器服务之间的异构性，使得基于 Docker 容器的应用在异构云 Kubernetes 集群之间的迁移成为可能。并在三层模型的基础上，本文提出了基于镜像预同步的应用迁移技术来优化基于 Docker 容器的应用跨异构容器云迁移的时间效率。下一步的工作应该将研究如何减少三层模型带来的网络带宽消耗，确保数据中心之间网络传输频繁时不会给用户正常工作带来影响。Docker 容器技术也在处在不断发展和更新的进程中，各种容器编排引擎和各云提供商的容器服务也在不断完善，力求充分发挥容器的优势，现阶段的研究无法囊括未来未知的各种情况，今后还需要继续关注 Docker 容器技术的发展，针对 Docker 容器技术的新特性进行完善。

参考文献：

[1] AbdelBaky M, Diaz-Montes J, Parashar M, *et al.* docker containers across multiple clouds and data centers [C]//Proc of the 8th IEEE/ACM International Conference on Utility and Cloud Computing. New York: ACM Press, 2015: 368-371.

[2] Kumar K M, Kumar D S, Murudi V. Multi Data Center Cloud Cluster Federation-Major Challenges & Emerging Solutions [C]//Proc of IEEE International Conference on Cloud Computing in Emerging Markets. Piscataway, NJ: IEEE Press, 2016: 107-122.

[3] Gupta V, Chaunhan S, Sharma D. Platform virtualization: understanding virtual machines, LXC, Docker, Kubernetes and Ubernetes [J]. International Journal of Innovations in Engineering and Technology. 2016, 7 (6): 442-447.

[4] Yu C Y, Huan F. Live migration of docker containers through logging and replay [C]//Proc of the 3rd International Conference on Mechatronics and Industrial Informatics. Paris, France: Atlantis Press, 2015: 623-626.

[5] Mirkin A, Kuznetsov A, Kolyshkin K. Containers checkpointing and

live migration [C]// Proc of Linux Symposium. 2008: 85-90.
[6] Laadan O, Hallyn S E. Linux-CR transparent application checkpoint-restart in Linux [C]// Proc of the Linux Symposium. 2010: 159-172.
[7] Machen A, Wang S Q, Leung K K, *et al.* Live service migration in mobile edge clouds [J]. IEEE Wireless Communications. 2018, 25 (2): 140-147.
[8] CRIU [EB/OL]. [2018-08-28]. https://criu.org/Usage_scenarios#Container_live_migration.
[9] 浙江大学 SEL 实验室. Docker 容器与容器云 [M]. 2 版. 北京: 人民邮电出版社, 2016.
[10] Kubernetes [EB/OL]. [2018-09-21]. https://kubernetes. io/.
[11] Docker Swarm [EB/OL]. [2018-09-21]. https://docs. docker. com/.
[12] 毛祺, 卢胜林. 基于 Docker Swarm 集群的容器迁移策略的实现 [J]. 信息技术, 2016 (9): 156-160. (Mao Qi, Lu Shenglin. Implementation of container migration strategy based on the docker swarm cluster [J]. Information Technology, 2016 (9): 156-160.)
[13] 卢胜林, 倪明, 张翰博. 基于 Docker Swarm 集群的调度策略优化 [J]. 信息技术, 2016 (7): 147-151. (Lu Shenglin, Ni Ming, Zhang Hanbo. The optimization of scheduling strategy based on the Docker swarm cluster [J]. Information Technology, 2016(7): 147-151.)